Dokumentáció - Kész

Programozói:

Programomban egy állatorvosi rendelő munkáját segítem. Számos könyvtárat igénybe vettem a standard könyvtárakon kívül a feladat megoldása során: string.h, stdbool.h. Előbbit a sztringkezelő (strcmp, strstr) függvények miatt, utóbbit az oltando_e függvényemben az igaz-hamis értékek visszaadásánál használtam ki, mint a visszaadott értéket megszabó feltétel. A sor módosításának kivitelezéséhez (modosit_adat függvény) definiáltam a MAX_HOSSZ konstanst 250 karakter hosszúságúra, ami az új sor beolvasásánál és az adatok vissza írattatásánál segítenek.

A főmenüvel kezdődik a program, amely a megadott funkciókat hivatott ismertetni és az elérésükhöz szükséges számjegyek lenyomásával elérhetővé válnak. Ezt követően a legtöbb függvénynél valamilyen adat bekérése következik. A switch-case szerkezetnek és a break parancsnak köszönhetően, ameddig a megfelelő számjegyet le nem nyomja a felhasználó (7, ami a kilépést jelenti a főmenüből), addig folyamatosan kihasználhatja a program funkcióit.

A megoldásom alapja egy egyszeresen láncolt lista ,amely dinamikusan van eltárolva, ami tartalmazza az állat nevét, faját, azonosítóját, az utolsó vizsgálatkor (közvetlenül az oltás előtt) megállapított egészségi állapotát, ami a Beteg/Egészséges tulajdonságot jelentheti, a tulajdonos vezeték-, keresztnevét és telefonszámát, valamint az adatszerkezetre jellemző *kov elemet, ami a következő listaelemre mutató pointert reprezentálja és fontos szerepet kap a ciklusokban, a kiiratatásnál és a beolvasott listával kapcsolatos műveleteknél.

A függvények, amiket használok:

A lista_beolvas() függvény a teljes_lista (összes beolvasott elem által alkotott lista) visszatérési értékével (pontosabban az erre mutató pointerrel tér vissza), amely paraméterként kapja a fájlra mutató *fp pointert és a megnyitandó, kezelendő függvény nevét. Létrehozok egy segédstruktúrát a stacken, hogy a beolvasott adatokat ebbe ideiglenesen el tudjam tárolni, majd strcpy() vagy egyenlőség operátort használva bemásolom őket, az új Allatok *uj dinamikusan foglalt listaelembe (struktúrába). A helyfoglalás és másolás műveletek minden alkalommal végrehajtódnak, amikor a while feltételében lévő fscanf() függvény mind a 10 várt értéket sikeresen beolvassa, amikor a fájl egész tartalma (ami eleget tesz a formátumnak) be lett olvasva és el lett tárolva, akkor a teljes_lista-ban benne lesz minden listaelem. A fájl bezáródik a függvény végén.

A listakiir() függvény, amely paraméterként kapja a kírandó listát (jelen esetben ez az előzőleg beolvasott és eltárolt) és bejárja a listát elemről-elemre a for ciklus segítségével és kiírja az elemeket a megadott formátumban. Az elemek kiiratása, úgy történik, hogy mindig a legutoljára bevitt elem van legfelül a kiiratásnál.

A kereses 3-féle módon, függvénnyel oldható meg. Minden esetben a paraméterként kapott listát bejárják egy for ciklussal, addig ameddig az aktuális elem NULL pointer nem lesz. A léptetést az eleje=eleje->kov kifejezés biztosítja. Az állat neve mindig szerepel a keresett adatpárokban, ugyanakkor mindig más elemmel: tulajdonos neve, telefonszam, faj. Az adatpárokkal való keresés során azt tapasztaltam, hogy nagyon pontosan lehet ily módon keresni a listaelemek között. Az if()-ben szereplő függvény minden esetben az strcmp() ,‘ÉS’

kapcsolatba hozva a két (vagy a tulajdonos nevénél 3) strcmp() -t. A függvények void típusúak, így a kiiratása a keresett elemnek a fő szerepük.

A bevisz_adat() függvény hasonlóan működik a beolvas()-hoz, mivel itt is segédváltozókat alkalmazok a fájlba beíráshoz, a megnyitás módja "at+", amivel biztosítom, hogy ha nem létezik a fájl, akkor hozza létre. A paraméterek az állat azonosítója (amit az azonosito_gen() függvény generál: megnézi, hány db listaelem van és a végén hozzáad 2-t és ezt az értéket adja vissza), a fájl neve. A megadott, új listaelemet, dinamikusan tárolom el, és itt is bemásolódnak a listaelem változóiba a bevitt adatok. Ezután fprintf()-el a megadott formátumban bekerül a fájlba az új elem, végül felszabadítódik a heap-ről a fájl bezárását követően.

A kiir_oltando() függvény azokat az állatokat irattatja ki, akiknek már esedékes az újbóli beoltása. Ehhez egy bool típusú értéket visszaadó függvényt használok, ami a paraméterként kapott ev, honap, nap felhasználó által megadott aktuális dátum segítségével meghatározza, hogy a listában lévő állatok közül kit kell beoltani. Ekkor true-t ad vissza, ha nem esedékes még, akkor false-t. A megadott formátumban, tulajdonos szerint írja ki az érintett állatokat, sorokba törve az adatokat, így jobban átláthatóbbá válik az adott listaelem.

A visszair() függvény a paraméterként kapott listát vissza írja a fájlba, törölve az előző tartalmát.

Az allat_modosit() függvény paraméterként kapja módosítandó listát és az állat nevét, a tulajdonos telefonszámát, ami alapján megkeresi a módosítandó elemet a lista bejárása során. Ha az if()-ben szereplő strcmp()-k egyezést adnak vissza, akkor az Allatok ideiglenes struktúrába beolvasott adatokat bemásolja az *uj dinamikusan tárolt listaelembe, amit be fog másolni az érintett módosítandó elembe. Ezt követően meghívódik a visszair() függvény és lementi a fájlba a módosított listát.

A torol_adat() void típusú függvény, ami az érintett, feltételnek eleget tevő listaelemet törli, kiiratja a listát és visszaírja a fájlba. Mind a törlésnél és a módosításnál is a keresési paraméterek az állat neve és tulajdonos telefonszáma változók, ami specifikus minden állat esetén.

A lista_felszabadit() függvény rekurzív módon szabadítja fel a dinamikusan tárolt listaelemeket, kezdve először a *kov (következő listaelemre mutató pointerrel), majd, az ezután már nyugodtan felszabadítható listaelemmel (arra mutató pointerrel).

A fuggvenyek1.h header fájl tartalmazza a függvényeket és a dinamikus struktúrát.

Felhasználói:

A felhasználónak van egy fontos dolga: pontosan kell megadnia a keresendő, beírandó adatokat, így várható a sikeres keresés és a pontos egyezés megtalálása. Ennek módja a megfelelő funkció igénybevételekor ki van írva. A bevitt szám/szöveg/karakter bevitele után itt is Enter megnyomásával indíthatja el a felhasználó az adott programrészt/menüpontot. A dátum bevitelekor alkalmazni kell az következő formátumot: pl. 2020 9 30. A program menü vezérelt, tehát a megfelelő karakterek beütésével és az Enter megnyomásával a menüpontok közül választhat a felhasználó és adott esetben, ha végzett a feladatával, akkor ki tud lépni a programból. Figyelnie kell a felhasználónak arra, hogy a bizonyos adatok hossza ne halladja meg a 29 karakter hosszúságot.

Elnézést kérek a késve leadásért, problémába ütköztem a feladat bizonyos pontjainak megoldásakor.